Deep Learning based Residual Attention Network for Malware Detection in Cyber Security

[1]Dr.P.Yejdani Khan M.Tech(ph.d) [2]K.Yedukondalu
[1]CSE, NEC., Gudur
2Assosiate Professor,CSE Department, NEC., Gudur

Abstract: Malware is malicious software disseminated to infiltrate the secrecy, integrity, and functionality of a system, such as viruses, worms, Trojans, backdoors, and spyware. With computers and the Internet being essential in everyday life, malware poses a serious threat to their security. To defend against an increasing number of sophisticated malware attacks, deep-learning based Malware Detection Systems (MDSs) have become a vital component of our economic and national security. The dataset, malware dataset is implemented as input. The input dataset is taken from dataset repository. Based on the characteristics of the observations, the dataset was created in a UNIX / Lunix-based virtual machine for classification purposes, which are harmless with malware software for Android devices. The data set consists of 100,000 observation data and 35 features. In our process, we have to implement the machine learning algorithms such as random forest and Logistic regression. After that, the results shows that the accuracy, precision, recall, f1-score.

Keywords: *component; formatting; style; styling; insert*

l.INTRODUCTION

Malware is malicious software disseminated to infiltrate the secrecy, integrity, and functionality of a system, such as viruses, worms, Trojans, backdoors, and spyware. With computers and the Internet being essential in everyday life, malware poses a serious threat to their security. It is estimated that one in four computers operating in the U.S. are infected with malware. The threat malware poses is not only emotional, but financial as well. According to a recent report from Kaspersky Lab, up to one billion dollars was stolen in roughly two years from financial institutions worldwide, due to malware attacks.

of As a result, the detection of malware is of major concern to both the anti-malware industry and researchers. In order to protect legitimate users from the attacks, the majority of anti-malware software products (e.g., Comodo, Symantec, and Kaspersky) use the signature-based method of detection .Signature is a short string of bytes, which is unique for each known malware so that its future examples can be correctly classified with a small error rate. However, this method can be easily evaded by malware attackers through the techniques such as encryption, polymorphism and

obfuscation. Furthermore, malicious files are being disseminated at a rate thousands per day, making it difficult for this signature-based method to be effective.

In order to combat the malware attacks, intelligent malware detection techniques need to be investigated. As a result, many researches have been conducted on intelligent malware detection by applying data mining and machine learning techniques in recent years. Malware, short for Malicious Software, is growing continuously in numbers and sophistication as our digital world continuous to grow. It is a very serious problem and many efforts are devoted to malware detection in today's cyber security world. Many machine learning algorithms are used for the automatic detection of malware in recent years. Most recently, deep learning is being used with better performance..

II.LITERATURE

The increasing popularity of Android apps attracted widespread attention from malware authors. Traditional malware detection systems suffer from some shortcomings; computationally expensive, insufficient performance or not robust enough. To address this challenge, we build a novel and highly reliable deep learning framework, named AMalNet, to learn multiple embedding representations for Android malware detection and family attribution, introduce a version of Graph Convolutional Networks for modeling high-level graphical semantics, which automatically identifies and learns the semantic and sequential patterns, use an Independently Recurrent Neural Network to decode the deep semantic information, making full use of remote dependent information between nodes to independently extract features. The experimental results on multiple benchmark datasets indicated that the AMalNet framework outperforms other state-of-the-art techniques significantly.

Detection

In this paper, based on the Windows Application Programming Interface (API) calls extracted from the Portable Executable (PE) files, we study how a deep learning architecture using the stacked Auto Encoders (SAEs) model can be designed for intelligent malware detection. The SAEs model performs as a greedy layer wise training operation for unsupervised feature learning, followed by supervised parameter fine-tuning (e.g., weights and offset vectors). To the best of our knowledge, this is the first work that deep learning using the SAEs model based on Windows API calls is investigated in malware detection for real industrial application.

III. SYSTEM IMPLEMENTATION

MODULES:

- Data selection
- Data preprocessing

- Feature selection
- Data Splitting
- Classification
- Result Generation

MODULE DESCRIPTION

Input Data:

- The input data was collected from dataset repository.
- The data selection is the process of selecting the data for detecting the malware.
- In this project, we have to use the malware detection dataset
- The dataset which contains the information about the classification(malware and benign) ,host etc.,
- In python, we have to read the dataset by using the pandas packages.
- Our dataset, is in the form of '.csv' file extension.

Preprocessing:

- Data pre-processing is the process of removing the unwanted data from the dataset.
- Pre-processing data transformation operations are used to transform the dataset into a structure suitable for machine learning.
- Missing data removal: In this process, the null values such as missing values and Nan values are replaced by 0.
- Encoding Categorical data: That categorical data is defined as variables with a finite set of label values.

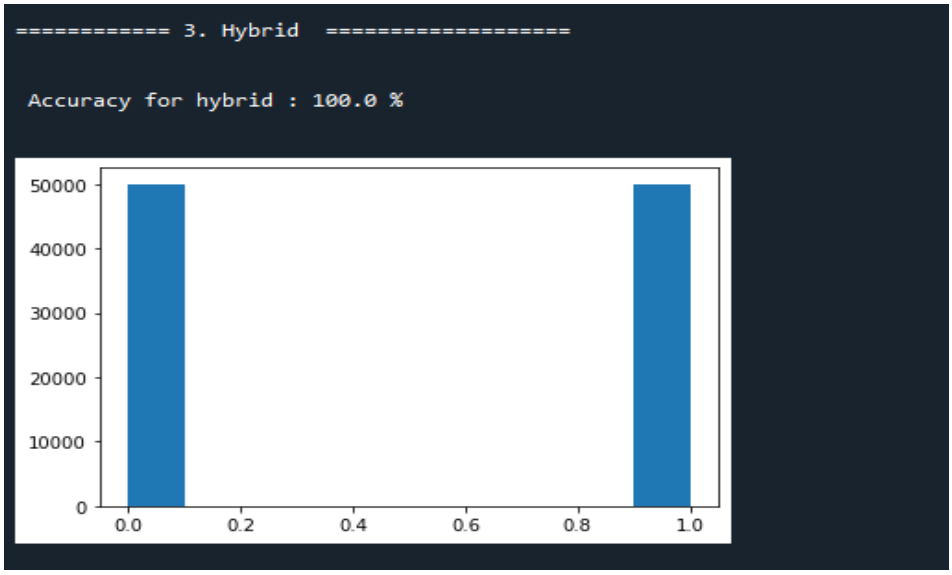
IV. SYSTEMTESTING

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. . A series of tests are performed before the system is ready for the user acceptance testing. Any engineered product can be tested in one of the following ways.

```
========================= Input Data ============================

                                                hash  ...  signal_nvcsw
0   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
1   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
2   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
3   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
4   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
5   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
6   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
7   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
8   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
9   42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
10  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
11  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
12  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
13  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
14  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
15  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
16  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
17  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
18  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
19  42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...  ...             0
```

```
[1] Malware

============================

[2]  Benign

============================

[3] Malware

============================

[4] Malware

============================

[5] Malware

============================

[6]  Benign

============================

[7] Malware

----------------------------
```



V.REFERENCES

- M. Alazab, S. Venkatraman, P. Watters, M. Alazab, and A. Alazab, ‘‘Cybercrime: The case of obfuscated malware,’’ in Global Security, Safety and Sustainability & e-Democracy (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering), vol. 99, C. K. Georgiadis, H. Jahankhani, E. Pimenidis, R. Bashroush, and A. Al-Nemrat, Eds. Berlin, Germany: Springer, 2012.
- M. Alazab, ‘‘Profiling and classifying the behavior of malicious codes,’’ J. Syst. Softw., vol. 100, pp. 91–102, Feb. 2015.
- S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, and J. Yearwood, ‘‘Hybrids of support vector machine wrapper and filter based framework for malware detection,’’ Future Gener. Comput. Syst., vol. 55, pp. 376–390, Feb. 2016.
- E. Raff, J. Sylvester, and C. Nicholas, ‘‘Learning the PE header, malware detection with minimal domain knowledge,’’ in Proc. 10th ACM Workshop Artif. Intell. Secur. New York, NY, USA: ACM, Nov. 2017, pp. 121–132.
- C. Rossow, et al., ‘‘Prudent practices for designing malware experiments: Status quo and outlook,’’ in Proc. IEEE Symp. Secur. Privacy (SP), Mar. 2012, pp. 65–79. [11] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. Nicholas. (2017). ‘‘Malware detection by eating a whole exe.’’
- L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, ‘‘Malware images: Visualization and automatic classification,’’ in Proc. 8th Int. Symp. Vis. Cyber Secur. New York, NY, USA: ACM, Jul. 2011, p. 4